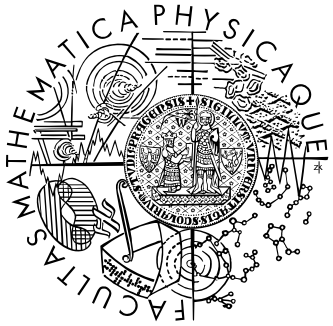
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Michael Grafnetter

Editor Relax NG

Katedra softwarového inženýrství

Vedoucí bakalářské práce: Mgr. Martin Nečaský

Studijní program: Informatika, Programování

2008

Na tomto mieste by som chcel poďakovať vedúcemu bakalárskej práce Mgr. Martinovi Nečaskému za cenné konzultácie a trpezlivosť. Moja vrelá vďaka patrí aj rodine, ktorá ma počas štúdia podporovala.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím s jej zapožičiavaním a zverejňovaním.

V Prahe dňa 6. augusta 2008 Michael Grafnetter

Obsah

Názov práce: Editor Relax NG

Autor: Michael Grafnetter (michael@matfyz.cz)

Katedra: Katedra softwarového inženýrství

Vedúci práce: Mgr. Martin Nečaský (martin.necasky@mff.cuni.cz)

Abstrakt: Cieľom predloženej bakalárskej práce bolo pokryť problematiku editorov jazyka Relax NG. Práca poskytuje nielen prehľad dostupných nástrojov a ich možností, ale ponúka aj ucelenú analýzu samotného jazyka Relax NG a jeho porovnanie s príbuznými jazykmi. Keďže pri mapovaní trhu nebol nájdený žiaden pokročilý bezplatný editor XML s podporou Relax NG, zámerom práce bolo aj vytvorenie takéhoto editoru. Program bol z dôvodu modularity navrhnutý ako open-source plug-in pre vývojové prostredie Eclipse a ponúka tak základ pre ďalšie rozširovanie jeho funkcionality komunitou.

Kľúčové slová: XML, editor Relax NG, XML schema, Eclipse

Title: Relax NG editor

Author: Michael Grafnetter (michael@matfyz.cz)

Department: Department of software engineering

Supervisor: Mgr. Martin Nečaský (martin.necasky@mff.cuni.cz)

Abstract: This thesis aims to cover the area of Relax NG editors. The work offers not only an overview of available tools and their abilities, but also gives a comprehensive analysis of the Relax NG language and comparison to similar languages. As no such free advanced editor had been found, the work also focused on creating one. The program has been developed as an open-source plug-in for the Eclipse Platform for modularity reasons and therefore provides a basis for further extending its functionality by the community.

Keywords: XML, Relax NG editor, XML schema, Eclipse

Kapitola 1
Úvod

1.1 Motivácia

Jazyk XML (eXtensible Markup Language) je jednoduchý, no zato veľmi flexibilný textový formát navrhnutý na zdieľanie štruktúrovaných dát a metadát. Svoje uplatnenie si našiel v dokumentovo aj dátovo orientovaných aplikáciách a používa sa najmä vo webovom a B2B (business-to-business) prostredí.

Štandard XML definuje iba syntax jazyka. Pridaním sémantických a integritných obmedzení vznikajú aplikačne špecifické jazyky založené na XML. Medzi najznámejšie z nich patria:

- XHTML – striktnejší variant HTML
- MathML – slúži na zápis matematických vzorcov
- ODF – používaný na ukladanie kancelárskych dokumentov
- SVG – vektorový grafický formát
- RSS – rodina formátov používaná na indikáciu (nového) obsahu
- SOAP – webové služby

Na definíciu sémantických pravidiel jazykov postavených na XML, ktoré sa nazývajú XML schéma, vznikli tiež špeciálne jazyky. Medzi najznámejšie z nich patria DTD a XML Schema z dielne konzorcia W3C. Menej známym jazykom z tejto rodiny je Relax NG, ktorý však oproti ostatným dovoľuje definovať vysoko úrovňové pravidlá. Lepšie tak môže odzrkadľovať predstavy dizajnéra daného jazyka, bez nutnosti pristupovať ku kompromisným riešeniam.

Jazyku Relax NG ale chýba dostatočná podpora v nástrojoch používaných na úpravu XML. Jedným z cieľov tejto práce je tak pridať túto podporu do prostredia Eclipse. To je jedným z najpopulárnejších nástrojov pre vývoj Java a webových aplikácií, v ktorých je jazyk XML často používaný.

1.2 Štruktúra práce

Text bakalárskej práce je rozdelený do siedmych kapitol. Druhá kapitola obsahuje úvod do problematiky XML schém a validácie XML. V tretej kapitole je na príkladoch popísaná syntax jazyka Relax NG a následne sú jeho možnosti porovnané s alternatívnymi jazykmi. Štvrtá kapitola sa zameriava na prieskum súčasného stavu editorov Relax NG. V piatej kapitole je obsiahnutá dokumentácia k vlastnému editoru Relax NG a šiesta kapitola pojednáva o implementačných detailoch. Záverečná siedma kapitola hodnotí prínos práce.

Kapitola 2

XML schémy

Pred detailným popisom samotného jazyka Relax NG je vhodné sa pozrieť na to, aké sú funkcie XML schém všeobecne a čo sa od nich očakáva.

2.1 Definícia pojmov

Rozlišujeme dve úrovne korektnosti XML dokumentu:

- **Správne štruktúrovaný** (well-formed) dokument spĺňa všetky syntaktické pravidlá definované v štandarde XML. Dokument, ktorý nie je správne štruktúrovaný, sa nepovažuje za XML.
- **Validný** dokument, okrem toho, že je správne štruktúrovaný, má k sebe aj pridruženú schému a vyhovuje všetkým sémantickým požiadavkám v nej definovaných. Proces kontroly, či XML súbor zodpovedá svojej schéme, sa nazýva validácia.

Program, ktorý analyzuje štruktúru XML dokumentu, sa nazýva XML parser. Ak navyše pri spracúvaní dokumentu vykonáva aj jeho validáciu a hlási prípadné chyby, hovoríme o validujúcom parseri.

2.2 Správne štruktúrované dokumenty

Základná syntax XML je jasná z tohto príkladu:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- this is some comment -->
<message type="e-mail">
      <head>
            <to><![CDATA[ Peter<peter@domain.com> ]]></to>
            <from><![CDATA[ John<john@domain.com> ]]></from>
            <subject>Message</subject>
            <important />
      </head>
      <body>Message text.</body>
</message>
```

Aby bol dokument správne štruktúrovaný (well-formed), musí spĺňať aspoň nasledovné požiadavky:

- Musí mať práve jeden koreňový element (v príklade `<message>`). Ostatné elementy sú v ňom vnorené.
- Neprázdne elementy sú ohraničené začiatočným a koncovým tagom (`<head>` a `</head>`). Prázdne elementy môžu byť označené nepárovým (self-closing) tagom (`<important/>`, čo je ekvivalentné zápisu `<important></important>`).
- Všetky hodnoty atribútov (`type="e-mail"`) sú ohraničené buď jednoduchými (`'`), alebo dvojitými (`"`) úvodzovkami. Jednoduchá úvodzovka však musí byť uzavretá jednoduchou a dvojitá dvojitou. Každému výskytu hraničnej úvodzovky v hodnote musí predchádzať špeciálny znak (`/`).
- Tagy môžu byť vnorené (napríklad `<head>` v `<message>`), no nesmú sa prekrývať, teda každý synovský element musí byť ukončený pred rodičovským.
- Dokument musí byť v zhode so svojim deklarovaným kódovaním (`encoding="UTF-8"`), ktoré môže byť explicitne uvedené v deklarácii typu dokumentu, prípadne v HTTP hlavičke. Implicitným kódovaním je UTF-8.
- V názvoch elementov sa rozlišujú veľké a malé písmená, takže napríklad tagy `<Message>` a `</message>` netvoria odpovedajúci pár.

Exaktne definovanú syntax jazyka XML v Backus-Naurovej forme je možné nájsť v špecifikácii XML [9].

2.3 Validné dokumenty a XML schémy

Správne štruktúrované dokumenty sú považované za validné iba vtedy, ak spĺňajú požiadavky schémy, s ktorou sú asociované. Tieto požiadavky obvykle zahŕňajú nasledovné obmedzenia:

- Elementy alebo atribúty, ktoré môžu alebo musia byť v dokumente použité.
- Štruktúra elementov, obvykle určená nejakou formou regulárnych výrazov.
- Interpretácia znakových dát, napríklad významnosť bielych znakov, či konverzia reťazcov na iné dátové typy, napríklad čísla, dátumy či URL.

Mechanizmy asociovania XML dokumentu so schémou sú rôzne a líšia sa od jazyka k jazyku. Asociácia môže byť dosiahnutá priamo odkazom v XML dokumente, alebo nejakým externým spôsobom ponechaným na konkrétne implementácie.

XML schémy tak plnia viaceré dôležité funkcie:

- Sú používané ako štandard, voči ktorému môže byť dokument skontrolovaný, či mu vyhovuje. To zaručuje interoperabilitu medzi aplikáciami využívajúcimi ten istý štandard.
- Validácia sa v aplikáciách nachádza na úrovni ošetrenia vstupov.
- Slúžia ako návod pri vytváraní XML dokumentov.
- Môžu byť použité ako sada invariantov pri spracúvaní XML dokumentov.
- Poskytujú relatívne vysokú úroveň abstrakcie nad dokumentmi rovnakého typu.
- Editormi XML sú využívané k automatickému kontextovému dopĺňaniu elementov, atribútov a pod.

Kapitola 3

Jazyk Relax NG

3.1 Úvod

RELAX NG (Regular Language for XML Next Generation) je jednoduchý jazyk na popis XML schém postavený na jazykoch Relax a Trex. Definuje vzor štruktúry a obsahu XML dokumentu, čím jednoznačne vymedzuje triedu všetkých XML dokumentov, ktoré tomuto vzoru odpovedajú.

Vyvinuté boli dva druhy syntaxe. Tá originálna je postavená na XML, takže je pri jej spracovaní možné využiť všetky prístupy vyvinuté pre samotné XML. Neskôr vznikla aj kompaktná syntax, nezaložená na XML, ktorá je vhodnejšia na ručnú úpravu. Vzhľadom na dostupnosť 1:1 konverzných nástrojov je výber konkrétnej z nich iba vecou vkusu. Podľa neformálnej konvencie majú schémy napísané v XML syntaxi koncovku RNG a v kompaktnej syntaxi koncovku RNC.

Špecifikácia Relax NG bola vyvinutá pod záštitou konzorcia OASIS skupinou Relax NG Technical Commitee a v čase písania práce sa nachádzala vo finálnej fáze štandardizácie v rámci ISO/IEC 19757: Document Schema Definition Languages (DSDL) [17].

3.2 XML Syntax

Namiesto formálnej špecifikácie syntaxe, ktorá je voľne prístupná na stránkach organizácie OASIS [18], sú jednotlivé konštrukcie demonštrované na konkrétnych príkladoch a porovnávané s konkurenčným DTD. Všetky príklady sú voľne prevzaté z oficiálnej dokumentácie.

Na úvod treba upozorniť, že používaná terminológia nie je celkom jednoznačná, pretože gramatika aj jej inštancie sú písané v jednom a tom istom jazyku – XML. Napríklad pravdivé tvrdenie, že v Relax NG sa na zápis vzoru pre XML element používa element s názvom element, môže pôsobiť zmätene. Aby sa predišlo nedorozumeniam, sú v tomto texte slová element, ktorých význam je element s názvom

element v Relax NG schéme, vysádzané neproporčným písmom. Element schémy je zas nazývaný vzor (pattern).

3.2.1 Princípy

Vychádzajme z nasledovnej XML reprezentácie adresára e-mailových kontaktov:

```
<addressBook>
  <card>
    <name>John Smith</name>
    <email>js@example.com</email>
  </card>
  <card>
    <name>Fred Bloggs</name>
    <email>fb@example.net</email>
  </card>
</addressBook>
```

Príslušné DTD by vyzeralo takto:

```
<!DOCTYPE addressBook [
<!ELEMENT addressBook (card*)>
<!ELEMENT card (name, email)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
]>
```

Schéma v Relax NG by potom mohla vyzerať nasledovne:

```
<element name="addressBook"
xmlns="http://relaxng.org/ns/structure/1.0">
  <zeroOrMore>
    <element name="card">
      <element name="name">
        <text/>
      </element>
      <element name="email">
        <text/>
```

```
      </element>
    </element>
  </zeroOrMore>
</element>
```

Vzoru `text` vyhovuje ľubovoľný, aj prázdny text. Ak by mal byť element `addressBook` neprázdny, mohli by sme použiť `oneOrMore` namiesto `zeroOrMore`. Ďalej môžeme do každého záznamu pridať voliteľný element `note`.

```
<element name="addressBook"
xmlns="http://relaxng.org/ns/structure/1.0">
  <oneOrMore>
    <element name="card">
      <element name="name">
        <text/>
      </element>
      <element name="email">
        <text/>
      </element>
      <optional>
     <element name="note">
       <text/>
     </element>
      </optional>
    </element>
  </oneOrMore>
</element>
```

Všetky elementy špecifikujúce vzor sa musia nachádzať v mennom priestore s URI http://relaxng.org/ns/structure/1.0. Predošlé príklady deklarujú východzí menný priestor, no rovnako je možné použiť iný, špecifikáciou jeho prefixu:

```
<rng:element name="addressBook"
xmlns:rng="http://relaxng.org/ns/structure/1.0">
  <rng:zeroOrMore>
    <rng:element name="card">
      <rng:element name="name">
        <rng:text/>
      </rng:element>
      <rng:element name="email">
```

```
          <rng:text/>
        </rng:element>
      </rng:element>
    </rng:zeroOrMore>
</rng:element>
```

3.2.2 Výber

Pridajme do príkladu voliteľné rozdelenie mena na krstné meno a priezvisko:

```
<addressBook>
  <card>
    <givenName>John</givenName>
    <familyName>Smith</familyName>
    <email>js@example.com</email>
  </card>
  <card>
    <name>Fred Bloggs</name>
    <email>fb@example.net</email>
  </card>
</addressBook>
```

V schéme použijeme vzor `choice` a elementy `givenName` a `familyName` zlúčime do skupiny, aby sa chovali ako jeden celok:

```
<element name="addressBook">
  <zeroOrMore>
    <element name="card">
      <choice>
        <element name="name">
          <text/>
        </element>
        <group>
          <element name="givenName">
            <text/>
          </element>
          <element name="familyName">
            <text/>
```

```
            </element>
          </group>
        </choice>
        <element name="email">
          <text/>
        </element>
        <optional>
      <element name="note">
        <text/>
      </element>
        </optional>
    </element>
  </zeroOrMore>
</element>
```

Tomu by zodpovedalo toto DTD:

```
<!DOCTYPE addressBook [
<!ELEMENT addressBook (card*)>
<!ELEMENT card ((name | (givenName, familyName)), email, note?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT givenName (#PCDATA)>
<!ELEMENT familyName (#PCDATA)>
<!ELEMENT note (#PCDATA)>
]>
```

3.2.3 Atribúty

Obsah záznamu môžeme presunúť zo synovských elementov do atribútov:

```
<!DOCTYPE addressBook [
<!ELEMENT addressBook (card*)>
<!ELEMENT card EMPTY>
<!ATTLIST card
  name CDATA #REQUIRED
  email CDATA #REQUIRED>
]>
```

V Relax NG schéme stačí iba zmeniť vzor element na `attribute`:

```
<element name="addressBook">
  <zeroOrMore>
    <element name="card">
      <attribute name="name">
        <text/>
      </attribute>
      <attribute name="email">
        <text/>
      </attribute>
    </element>
  </zeroOrMore>
</element>
```

Za pozornosť stojí fakt, že kým v prípade elementov na ich poradí záleží, poradie atribútov pri validácii nehrá rolu. Uvedenie deklarovaného atribútu je povinné, no voliteľnosť sa dá zabezpečiť vzorom `optional`:

```
<element name="addressBook">
  <zeroOrMore>
    <element name="card">
      <attribute name="name">
        <text/>
      </attribute>
      <attribute name="email">
        <text/>
      </attribute>
      <optional>
        <attribute name="note">
          <text/>
        </attribute>
      </optional>
    </element>
  </zeroOrMore>
</element>
```

Na atribúty je rovnako ako u elementov možné aplikovať vzory `choice` a `group`, dokonca sa môžu neobmedzene miešať s `elementami`:

```
<element name="addressBook">
  <zeroOrMore>
    <element name="card">
      <choice>
     <element name="name">
       <text/>
     </element>
     <attribute name="name">
       <text/>
     </attribute>
      </choice>
      <choice>
     <element name="email">
       <text/>
     </element>
     <attribute name="email">
       <text/>
     </attribute>
      </choice>
    </element>
  </zeroOrMore>
</element>
```

Uvedenej schéme by zodpovedali aj tieto inštancie:

```
<card name="John Smith" email="js@example.com"/>
<card email="js@example.com" name="John Smith"/>
<card email="js@example.com"><name>John Smith</name></card>
<card name="John Smith"><email>js@example.com</email></card>
<card><name>John Smith</name><email>js@example.com</email></card>
```

Kvôli nesprávnemu poradiu elementov by ale už nevyhovovala táto:

```
<card><email>js@example.com</email><name>John Smith</name></card>
```

Čo sa týka východzieho typu hodnoty atribútu, je konštrukcia

```
<attribute name="email"/>
```

iba skratkou za

```
<attribute name="email">
  <text/>
</attribute>
```

Ak má byť element bez atribútov aj synovských elementov, je nutné ako jeho obsah explicitne uviesť `<empty/>`.

3.2.4 Pomenované vzory

Na predošlých príkladoch je vidno, že XML syntax je veľmi „upísaná“ a preto je vhodné zložitejšie a často používané vzory pomenovať a ďalej sa na ne odkazovať referenciami. Namiesto

```
<element name="addressBook">
  <zeroOrMore>
    <element name="card">
      <element name="name">
     <text/>
      </element>
      <element name="email">
        <text/>
      </element>
    </element>
  </zeroOrMore>
</element>
```

môžeme napísať

```
<grammar>

  <start>
    <element name="addressBook">
      <zeroOrMore>
     <element name="card">
       <ref name="cardContent"/>
     </element>
      </zeroOrMore>
    </element>
```

```
  </start>

  <define name="cardContent">
    <element name="name">
      <text/>
    </element>
    <element name="email">
      <text/>
    </element>
  </define>

</grammar>
```

Element `grammar` sa skladá z jedného hlavného elementu `<start>`, ktorý definuje vzor pre `grammar` a z ľubovoľného počtu elementov `define`, na ktoré môžu viesť referencie. Povolené sú aj rekurzívne referencie, ale samozrejme môžu byť iba v rámci elementov:

```
<define name="inline">
  <zeroOrMore>
    <choice>
      <text/>
      <element name="bold">
        <ref name="inline"/>
      </element>
      <element name="span">
        <optional>
          <attribute name="style"/>
        </optional>
        <ref name="inline"/>
      </element>
    </choice>
  </zeroOrMore>
</define>
```

Odkazovať sa dá aj na vzory v externých súboroch. Na tento účel slúži element `externalRef`, ktorého povinný atribút `href` obsahuje URI súboru so vzorom:

```
<element name="note">
```

```
  <externalRef href="inline.rng"/>
</element>
```

Chovanie externých referencií je podobné ako u direktívy `include` v jazyku C. Vzory `grammar` je dovolené do seba aj vnárať. Výskyt vzoru `ref` potom odkazuje na definíciu z najbližšieho predka `grammar`.

3.2.5 Dátové typy

Vstavaná knižnica dátových typov Relax NG poskytuje iba základné typy `string` a `token`. Vzory sa ale môžu odkazovať aj na externe definované dátové typy. Ich podpora závisí od konkrétnej implementácie, no väčšina z nich si poradí s dátovými typmi z W3C XML Schema. Na identifikáciu reťazca, ktorý reprezentuje hodnotu pomenovaného typu, sa používa vzor `data`:

```
<element name="point"
datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
  <element name="x">
    <data type="double"/>
  </element>
  <element name="y">
    <data type="double"/>
  </element>
</element>
```

Z praktických dôvodov je atribút `datatypeLibrary` dedičný, aby ho nebolo nutné uvádzať pre každý element `data` zvlášť. Vo väčšine prípadov sa tak definuje už v koreňovom elemente. Ak sa v obsahu elementu nachádza vzor `data`, môže byť kombinovaný iba so vzorom `attribute`. Výnimkou z tohto pravidla je len vzor text. Nasledujúci fragment je preto chybný:

```
<element name="bad">
  <data type="int"/>
  <element name="note">
    <text/>
  </element>
</element>
```

Dátové typy môžu byť parametrizované. Dá sa tak nastaviť napríklad maximálna dĺžka reťazca či maximálna hodnota čísla:

```
<element name="email">
  <data type="string">
    <param name="maxLength">127</param>
  </data>
</element>
```

3.2.6 Vymenovaný typ

Vymenovaný typ je vhodný, ak chceme prípustné hodnoty atribútov obmedziť na malú konečnú množinu:

```
<element name="card">
  <attribute name="name"/>
  <attribute name="email"/>
  <attribute name="preferredFormat">
    <choice>
      <value>html</value>
      <value>text</value>
    </choice>
  </attribute>
</element>
```

V jazyku DTD by to vyzeralo takto:

```
<!DOCTYPE card [
<!ELEMENT card EMPTY>
<!ATTLIST card
  name CDATA #REQUIRED
  email CDATA #REQUIRED
  preferredFormat (html|text) #REQUIRED>
]>
```

Na rozdiel od DTD je ale možné obmedziť nielen hodnoty atribútov, ale aj obsah elementov. Reťazce sa s výnimkou sekcií `CDATA` porovnávajú až po normalizácii

oboch reťazcov, teda odrezania počiatočných i koncových bielych znakov a kontrakcie sekvencií bielych znakov do jednej medzery.

3.2.7 Zoznamy

Vzor `list` slúži na definovanie postupnosti tokenov oddelených bielymi znakmi. Pri validácii príde k rozbitiu reťazca na zoznam tokenov a ten je potom porovnávaný so vzorom.

Vzor elementu <`path`>, ktorý obsahuje postupnosť dvojíc reálnych čísel (teda párny počet), vyzerá takto:

```
<element name="path">
  <list>
    <oneOrMore>
      <data type="double"/>
      <data type="double"/>
    </oneOrMore>
  </list>
</element>
```

3.2.8 Prekladanie

Vo všetkých predchádzajúcich príkladoch záviselo na poradí elementov, čo môže byť pri ručnej tvorbe XML kontraproduktívne. Problém ale rieši vzor `interleave`, ktorého synovské elementy sa môžu v dokumente vyskytovať v ľubovoľnom poradí.

Typickým príkladom je hlavička XHTML, ktorá obsahuje práve jeden element <`title`>, najviac jeden element base a nula či viac výskytov elementov <`style`>, <`script`>, <`link`> a <`meta`>, v ľubovoľnej permutácii:

```
<define name="head">
  <element name="head">
    <interleave>
      <ref name="title"/>
      <optional>
        <ref name="base"/>
      </optional>
```

```
        <zeroOrMore>
          <ref name="style"/>
        </zeroOrMore>
        <zeroOrMore>
          <ref name="script"/>
        </zeroOrMore>
        <zeroOrMore>
          <ref name="link"/>
        </zeroOrMore>
        <zeroOrMore>
          <ref name="meta"/>
        </zeroOrMore>
      </interleave>
    </element>
</define>
```

Často sa vyskytujúcim prípadom je zmiešaný obsah, v ktorom je prípustné miešanie textu s nejakým vzorom (označíme ho `p`):

```
<interleave> <text/> p </interleave>
```

Má aj syntaktickú skratku s ekvivalentným významom:

```
<mixed> p </mixed>
```

3.2.9 Kombinovanie definícií

Ak gramatika obsahuje viacero definícií s rovnakým názvom, je nutné v atribúte `combine` uviesť, aký spôsobom sa majú zlúčiť do jednej. Na výber sú `choice` a `interleave`. Napríklad význam fragmentu

```
<define name="inline.class" combine="choice">
  <element name="bold">
    <ref name="inline"/>
  </element>
</define>

<define name="inline.class" combine="choice">
```

```
   <element name="italic">
     <ref name="inline"/>
   </element>
</define>
```

je zhodný s

```
<define name="inline.class">
   <choice>
     <element name="bold">
       <ref name="inline"/>
     </element>
     <element name="italic">
       <ref name="inline"/>
     </element>
   </choice>
</define>
```

Poradie definícií nie je podstatné. Dve definície so zhodným názvom musia mať nastavenú aj tú istú hodnotu `combine`. Kombinácia atribútov cez `interleave` má rovnaký efekt ako `group`. Analogicky sa dajú kombinovať aj viacnásobné výskyty elementu `start`.

Časté je aj použitie prázdnych definícií:

```
<define name="inline.extra">
     <notAllowed/>
</define>
```

Uplatnenie si nájdu pri kombinácií definícií (atribút `combine="choice"`) z viacerých súborov schémy. Vzor `notAllowed` tu má podobný význam ako `empty` u `group`.

3.2.10 Menné priestory

Keďže Relax NG zohľadňuje menné priestory, názvy elementov i atribútov sa skladajú z ich lokálneho mena a URI menného priestoru.

Na určenie URI menného priestoru má vzor element atribút `ns`. Napríklad vzoru

```
<element name="foo" ns="http://www.example.com">
  <empty/>
</element>
```

budú vyhovovať aj elementy

```
<foo xmlns="http://www.example.com"/>
<e:foo xmlns:e="http://www.example.com"/>
<example:foo xmlns:example="http://www.example.com"/>
```

no nie elementy

```
<foo/>
<e:foo xmlns:e="http://WWW.EXAMPLE.COM"/>
<example:foo xmlns:example="http://www.example.net"/>
```

Prázdna hodnota atribútu `ns` znamená nulové resp. absentujúce URI, čo je aj východzia hodnota. Aby sa predišlo prípadným chybám pri častom písaní URI, je atribút `ns` dedičný. Ak teda element nemá uvedený explicitne uvedený menný priestor, implicitne sa použije menný priestor jeho najbližšieho predka. Na základe odporúčania W3C to však neplatí pre atribúty, kde neuvedenie `ns` rovno znamená prázdne URI. Obdobne funguje dedičnosť aj u kvalifikovaných mien. Napríklad

```
<element name="ab:addressBook"
xmlns:ab="http://www.example.com/addressBook"

xmlns:a="http://www.example.com/address">
  <zeroOrMore>
    <element name="ab:card">
      <element name="a:name">
        <text/>
      </element>
      <element name="a:email">
        <text/>
      </element>
    </element>
  </zeroOrMore>
</element>
```

má rovnaký význam ako

```
<element name="addressBook" ns="http://www.example.com/addressBook">
  <zeroOrMore>
    <element name="card" ns="http://www.example.com/addressBook">
      <element name="name" ns="http://www.example.com/address">
        <text/>
      </element>
      <element name="email" ns="http://www.example.com/address">
        <text/>
      </element>
    </element>
  </zeroOrMore>
</element>
```

3.2.11 Menné triedy

Doteraz sme vždy špecifikovali konkrétne názvy elementov a atribútov. S použitím menných tried je ale možné definovať celú množinu názvov, aj nekonečnú. Najjednoduchšou triedou je `anyName`, ktorej vyhovuje každý názov bez ohľadu na URI menného priestoru. Schéma, voči ktorej sú validné všetky správne štruktúrované XML dokumenty, vyzerá takto:

```
<grammar>

  <start>
    <ref name="anyElement"/>
  </start>

  <define name="anyElement">
    <element>
      <anyName/>
      <zeroOrMore>
       <choice>
        <attribute>
          <anyName/>
        </attribute>
        <text/>
```

```
        <ref name="anyElement"/>
      </choice>
       </zeroOrMore>
    </element>
  </define>

</grammar>
```

Trieda `nsName` je prísnejšia, obsahuje všetky názvy z menného priestoru, ktorý je určený atribútom `ns`. Obe triedy môžu obsahovať klauzulu `except`, ktorou sa dá definovať rozdiel množín názvov:

```
<element name="card" ns="http://www.example.com">
  <zeroOrMore>
    <attribute>
      <anyName>
        <except>
          <nsName/>
          <nsName ns=""/>
        </except>
      </anyName>
    </attribute>
  </zeroOrMore>
  <text/>
</element>
```

Tento vzor tak spĺňajú všetky elementy s výnimkou priestoru vlastnom elementu <`card`>. U množinových operácií sa môže hodiť aj trieda `name`, ktorá obsahuje iba jeden konkrétny názov:

```
<name>xml:space</name>
```

3.3 Kompaktná syntax

Kompaktná syntax Relax NG nie je založená na XML, no dá sa doň prevádzať 1:1. Jedná sa o syntaktickú skratku, ktorá odstraňuje zbytočnú redundanciu.

3.3.1 Základné konštrukcie

Prvý príklad z predošlej kapitoly po transformácii do kompaktnej syntaxe vyzerá takto:

```
element addressBook {
  element card {
    element name { text },
    element email { text }
  }*
}
```

Namiesto vzorov sa teda používajú štandardné operátory známe aj z DTD, viď. tabuľka:

Vzor v XML syntaxi	Operátor v kompaktnej syntaxi
zeroOrMore	*
oneOrMore	+
optional	?
choice	|
group	( )
interleave	&

Obsah vzorov je ohraničený zloženými zátvorkami {} a jednoriadkové komentáre začínajú znakom # a pokračujú až do konca riadku.

3.3.2 Pomenované vzory

Syntax u pomenovaných vzorov tiež ničím neprekvapí:

```
grammar {
  start =
    element addressBook {
      element card { cardContent }*
    }
  cardContent =
    element name { text },
    element email { text }
}
```

U gramatík na najvyššej úrovni je dokonca možné otváraciu a zatváraciu sekvenciu `grammar { }` úplne vynechať. Externé referencie označuje kľúčové slovo `external`:

```
element addressBook {
  element card {
    element name { text },
    element email { text },
    element note { external "inline.rnc" }?
  }*
}
```

Definície sa kombinujú operátormi `|=` a `&=`, takže zápis

```
inline.class |= element bold { inline }
inline.class |= element italic { inline }
```

je totožný zápisu

```
inline.class =
  element bold { inline }
  | element italic { inline }
```

3.3.3 Dátové typy

Dátové typy z XML Schema sú importované automaticky, takže sa dajú rovno používať:

```
element number { xsd:integer }

element email {
  xsd:string { minLength = "6" maxLength = "127" }
}
```

Import z iných URI je tiež možný:

```
datatypes xs = "http://www.w3.org/2001/XMLSchema-datatypes"

element number { xs:integer }
```

3.3.4 Vymenovaný typ

Syntax je podobná ako u alternatívy:

```
element card {
  element name { text },
  element email { text },
  element preferredFormat { "html" | "text" }
}
```

3.3.5 Zoznamy

Vzor `list` sa len prepíše do už známej syntaxe:

```
element path {
  list { (xsd:double, xsd:double)+ }
}
```

3.3.6 Menné priestory

Deklarácie menných priestorov sa vyskytujú na začiatku súboru alebo pred vzorom:

```
namespace a = "http://www.example.com/address"
namespace ab = "http://www.example.com/addressBook"

element ab:addressBook {
  element ab:card {
    element a:name { text },
    element a:email { text }
  }*
}
```

Aj v kompaktnej syntaxi je nutné si dávať pozor na menný priestor atribútov, pretože

```
default namespace = "http://www.example.com/address"
```

```
element addressBook {
  element card {
    attribute name { text },
    attribute email { text }
  }*
}
```

má význam

```
namespace ab = "http://www.example.com/address"

element ab:addressBook {
  element ab:card {
    attribute name { text },
    attribute email { text }
  }*
}
```

3.3.7 Menné triedy

Znak * je preťažený aj do významu ľubovoľného názvu, teda `anyName`:

```
start = anyElement
anyElement =
  element * {
    (attribute * { text }
     | text
     | anyElement)*
  }
```

Trieda `ns:*` obsahuje všetky názvy v mennom priestore deklarovanom prefixom `ns`. O čosi zložitejší príklad demonštruje použitie operátorov – a | na rozdiel a zjednotenie menných tried:

```
start = anyElement
anyElement =
  element * {
    (attribute * { text }
```

```
    | text
    | anyElement)*
}
```

3.3.8 Reťazce

Reťazce ohraničené jednoduchými (') alebo dvojitými (") úvodzovkami nemôžu obsahovať znak nový riadok. Namiesto neho je možné použiť sekvenciu `\x{A}`, no kvôli pohodli boli zavedené aj trojité úvodzovky (''' a """) z jazyka Python, ktoré môžu obsahovať všetky literály. Reťazce sa spájajú operátorom ~.

3.4 Porovnanie Relax NG s DTD

Relax NG oproti DTD poskytuje navyše túto funkcionalitu:

- možnosť výberu medzi XML a kompaktnou syntaxou pri písaní schém
- podpora dátových typov
- rovnocenný prístup k elementom a atribútom
- podpora XML menných priestorov
- kontextovo senzitívny model
- podpora neusporiadaného obsahu

Tieto možnosti DTD v Relax NG chýbajú:

- deklarácia implicitných hodnôt atribútov
- validácia ID referencií
- entity
- notácie
- voľba významnosti bielych znakov

Existuje ale návrh rozšírení Relax NG DTD Compatibility [13], ktorý tieto nedostatky rieši a údajne by mal byť zapracovaný do budúcej verzie Relax NG.

Avšak najväčšou nevýhodou je nemožnosť k XML dokumentu priradiť schému v Relax NG. So situáciou sa tak autori každého validátora vyrovnávajú inak. Niektoré validátori za týmto účelom zavádzajú špeciálnu inštrukciu spracovania, v ktorej je uvedené URI schémy, iné zas vyžadujú cestu k schéme zadanú ako parameter.

3.5 Porovnanie Relax NG s XML Schema

3.5.1 Spoločné ciele

Oba tieto jazyky boli vyvinuté približne v rovnakom čase s cieľom nahradiť príliš obmedzujúce DTD. Obidvom sa podarilo nielen splniť hlavné ciele, teda použitie XML syntaxe, typovanie dát a podpora menných priestorov, ale aj značne rozšíriť vyjadrovacie možnosti schémy. Vďaka vysokému stupňu ich modularity sa dajú schémy rozdeľovať do viacerých súborov.

3.5.2 Výhody Relax NG oproti W3C XML Schema

- Pri návrhu Relax NG bola hlavným kritériom jednoduchosť a flexibilitu. Osoba zbehlá v XML bez problémov porozumie existujúcim Relax NG gramatikám a naučenie sa písať v Relax NG tiež nie je časovo náročné. To však neplatí o dokumentoch napísaných v XML Schema, ktoré sú náročnejšia na pochopenie a môžu v sebe ukrývať množstvo nekonzistencií, najmä u odvodzovania komplexných typov. Na druhú stranu, niekto môže preferovať väčšiu formálnosť a sémantický význam jednotlivých komponent.
- Špecifikácia štandardu XML Schema [9] je veľmi náročná na pochopenie. Navyše, odporúčanie XML Schema: Formal Description [8] je stále v pracovnom štádiu a neobsahuje úplný popis sémantiky. Preto sa naň nedá spoľahnúť ako na normatívny dokument. Špecifikácia Relax NG však obsahuje ucelený formálny popis sémantiky založený na teórii stromových automatov.
- Podpora atribútov v XML Schema je obdobná ako u DTD, teda veľmi obmedzená. Relax NG však medzi prístupom k atribútom a elementom nerobí žiadne rozdiely a umožňuje s nimi zachádzať rovnako:

```
<element name="user">
  <choice>
    <attribute name="has_name">
      <value>false</value>
    </attribute>
    <group>
      <attribute name="has_name">
        <value>true</value>
      </attribute>
```

```
        <element name="name"><text /></element>
      </group>
    </choice>
</element>
```

V tomto príklade je definovaný element user s povinným atribútom has_name. Ten môže nadobúdať iba hodnoty true a false a v prípade true musí element user obsahovať aj element name. V XML Schema sa takáto závislosť nadefinovať nedá.

- Relax NG povoľuje nedeterministický obsah:

```
<zeroOrMore>
  <ref name="odd" />
  <ref name="even" />
</zeroOrMore>
<optional>
  <ref name="odd" />
</optional>
```

Keď validátor narazí na niečo, čo zodpovedá vzoru `odd`, nie je známe, či sa jedná o voliteľnú poslednú referenciu `odd` alebo súčasť sekvencie `zeroOrMore`.

V XML Schema musia byť všetky postupnosti deterministické, takže mechanizmy ako je tento musia byť buď špecifikované inak, alebo úplne vynechané.

- V XML Schema neexistuje spôsob. ako špecifikovať, ktoré elementy môžu byť koreňové. Relax NG je v tomto ohľade jednoznačnejší.
- Oproti Relax NG poskytuje XML Schema slabšiu podporu neusporiadaného obsahu, napríklad sa v ňom musia vyskytnúť všetky synovské elementy.

3.5.3 Nevýhody Relax NG oproti XML Schema

- V Relax NG sa nedajú definovať východzie hodnoty atribútov.
- W3C XML Schema poskytuje atribút `xsi:schemaLocation`, ktorý určuje, na základe ktorej schémy sa má inštancia validovať. Relax NG týmto mechanizmom nedisponuje.

- Relax NG má vstavané iba dva dátové typy (`string` a `token`), no umožňuje import dátových typov aj z XML Schema, čo však musí podporovať daný validátor. Schémy v Relax NG tak nemusia byť medzi rôznymi parsermi prenosné.
- Na rozdiel od Relax NG umožňuje XML Schema definovať konkrétny počet alebo rozsah opakovaní vzoru. V Relax NG sa to dá docieliť iba veľmi prácne, násobným použitím vzoru. U vyšších čísel sa však jedná o prakticky nepoužiteľnú metódu.

XML Schema je aj napriek svojim mnohým nevýhodám oveľa známejšia a široko implementovaná v open-source aj komerčných XML parserov a editorov. Relax NG sa ale v poslednom čase dostáva čím ďalej tým viac do povedomia aj vďaka tomu, že bol zvolený ako primárny jazyk na popis schém pre populárne formáty, akými sú Docbook, smernice TEI a OpenDocument.

V prospech XML Schema hovorí najmä fakt, že pochádza z dielne W3C a je v hojnej miere používaná a podporovaná.

3.6 Relax NG ako medziformát

Vďaka svojej voľnosti sa Relax NG javí ako vhodný medziformát (pivot) pri konverzii medzi rôznymi schémovými jazykmi. Túto vlastnosť využíva napríklad nástroje Trang či Sun Multi-Schema Validator, ktoré okrem Relax NG podporujú aj pôvodný Relax, DTD a W3C XML Schema. Podľa autorov sa v Relax NG dá vyjadriť 99% obmedzení zapísaných v iných jazykoch.

Opačne to však neplatí, pretože v Relax NG sa dajú vytvoriť nepreložiteľné schémy. Našťastie je tento problém skôr teoretický, pretože v reálnych situáciách k nemu väčšinou nedochádza.

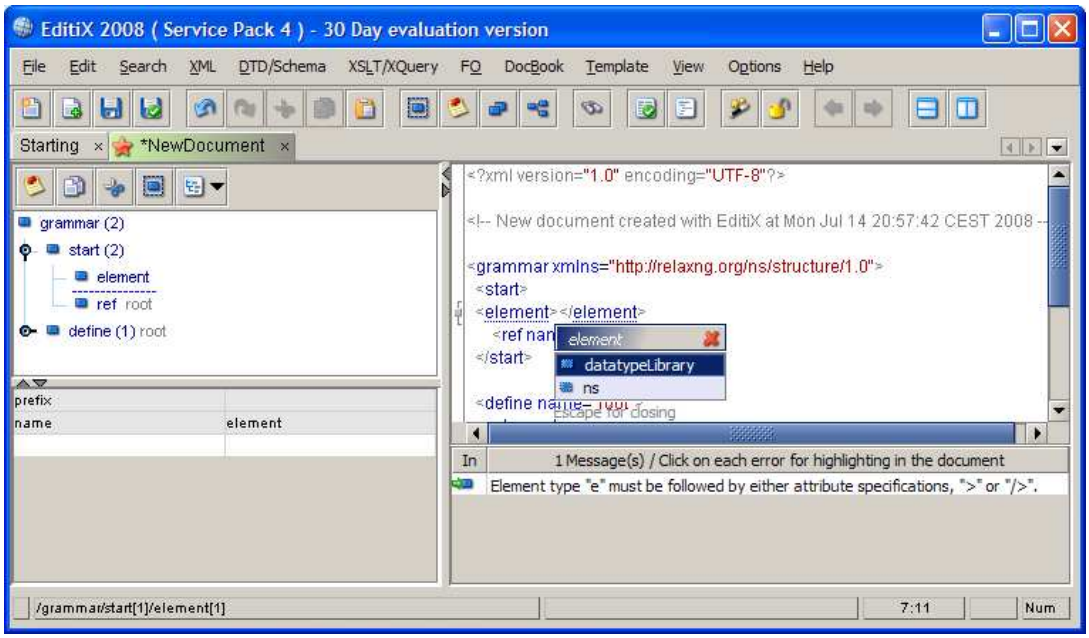
Kapitola 4

Editory podporujúce jazyk Relax NG

V tejto kapitole sa nachádza prehľad vlastností jednotlivých editorov s podporou jazyka Relax NG. Prieskum bol zameraný iba na programy v produkčnej fáze vývoja a z pochopiteľných dôvodov sa nedotýka všetkých takýchto editorov dostupných na trhu. Ich počet sa však odhadom pohybuje v ráde jednotiek a teda prehľad by mal zahŕňať väčšinu z nich, vrátane tých najpoužívanejších a odporúčaných na oficiálnych stránkach štandardu Relax NG.

4.1 EditX 2008

Tento multiplatformový proprietárny softvér ponúka okrem editoru zdrojového kódu s farebným zvýrazňovaním lexikálnych elementov a automatického dopĺňania slov aj pohľad na stromovú štruktúru XML stromu, ktorý je s editorom kódu dobre previazaný. Ďalšou funkciou je obojsmerná konverzia do formátov DTD a XML Schema. U editácie inštancií schém potom nechýba kontextové dopĺňanie a validácia. Schémy s kompaktnou syntaxou sa ale v programe otvoriť nedajú.



Obr. 1: Rozhranie EditX

4.2 XMLBuddy Pro 2.0

Z odľahčenej bezplatnej verzie program boli odstránené aj všetky nástroje na prácu s Relax NG a tvorca tohto softvéru neposkytuje funkčne neobmedzenú skúšobnú verziu. Nebolo tak možné program otestovať a nasledovné informácie sú preto založené iba na materiáloch od výrobcu.

Podpora pre Relax NG s kompaktnou syntaxou je ako u jedného z mála programov na rovnako vysokej úrovni ako pre Relax NG s XML syntaxou. Automatické dopĺňanie kódu, teda elementov, atribútov, hodnôt a entít na základe schémy je k dispozícii nielen u inštancií ale aj u samotných schém. Príkaz Open Definition, známy skôr z prostredí pre C-like jazyky, otvorí definíciu vybraného názvu v príslušnej schéme a netreba ju tak hľadať.

Pomocou dialógu sa dá k XML dokumentu priradiť schéma s patričným koreňovým elementom či menným priestorom, čo následne umožňuje validáciu. Tá prebieha na pozadí počas písania.

Zaujímavou sa javí možnosť vygenerovať schému na základe existujúceho XML dokumentu, čo môže značne zrýchliť vývoj schém a slúžiť ako štartovací bod pre ľudí, ktorí v jazyku Relax NG ešte nie sú úplne zbehlí. Tí, ktorí preferujú jeden jazyk pred druhým, uvítajú možnosť konverzie medzi schémami rôznych typov. Prácu urýchľujú aj sprievodcovia a šablóny pri vytváraní nových dokumentov, u ktorých je na výber z kódovania, menného priestoru, schémy a koreňového elementu.

Editor je k dispozícii nielen ako samostatná aplikácia, ale aj ako plug-in pre vývojové prostredie Eclipse.

4.3 oXygen XML Editor 9.3

Aj v prípade oXygen XML Editoru sa jedná o proprietárny platený, ktorého bezplatná verzia je orezaná o kľúčové vlastnosti, vrátane podpory Relax NG.

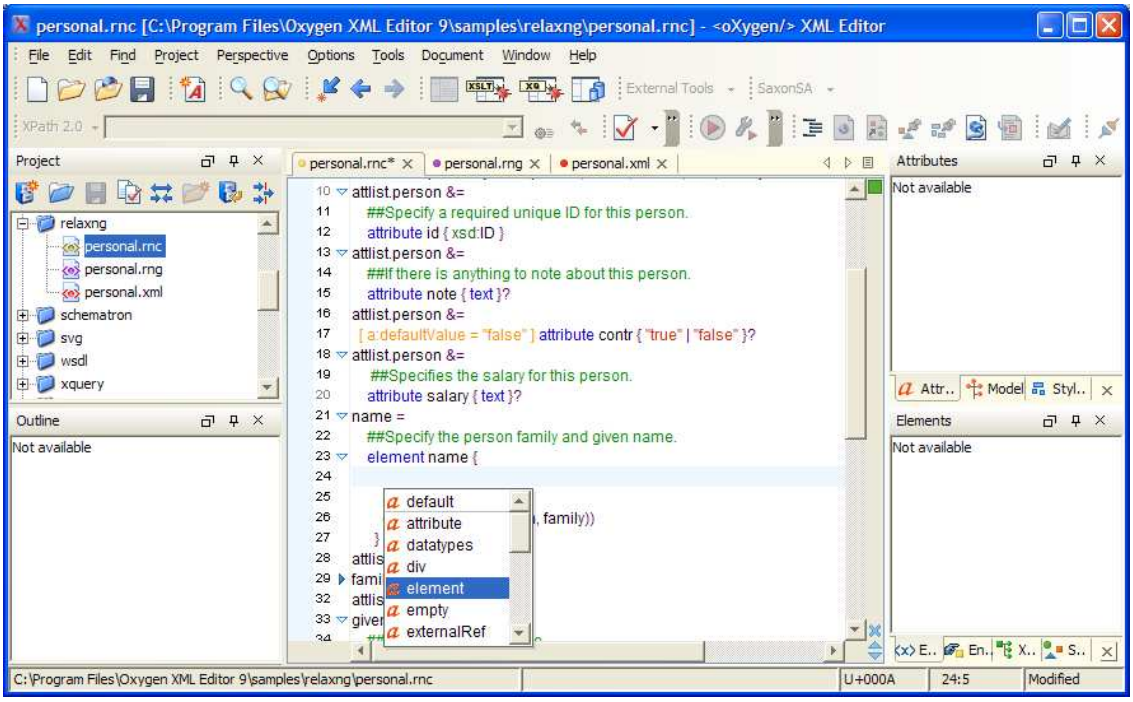
V edícii Enterprise je ale podpora Relax NG na vysokej úrovni. Textový editor poskytuje zvýrazňovanie, automatické doplňovanie kódu a skrývanie fragmentov pre XML aj kompaktnú syntax. Nechýba ani automatická a dávková validácia XML dokumentov na základe Relax NG. S priradením dokumentu k schéme sa autori vysporiadali implementáciou vlastnej inštrukcie spracovania:

```
<?oxygen RNGSchema="personal.rnc" type="compact"?>
```

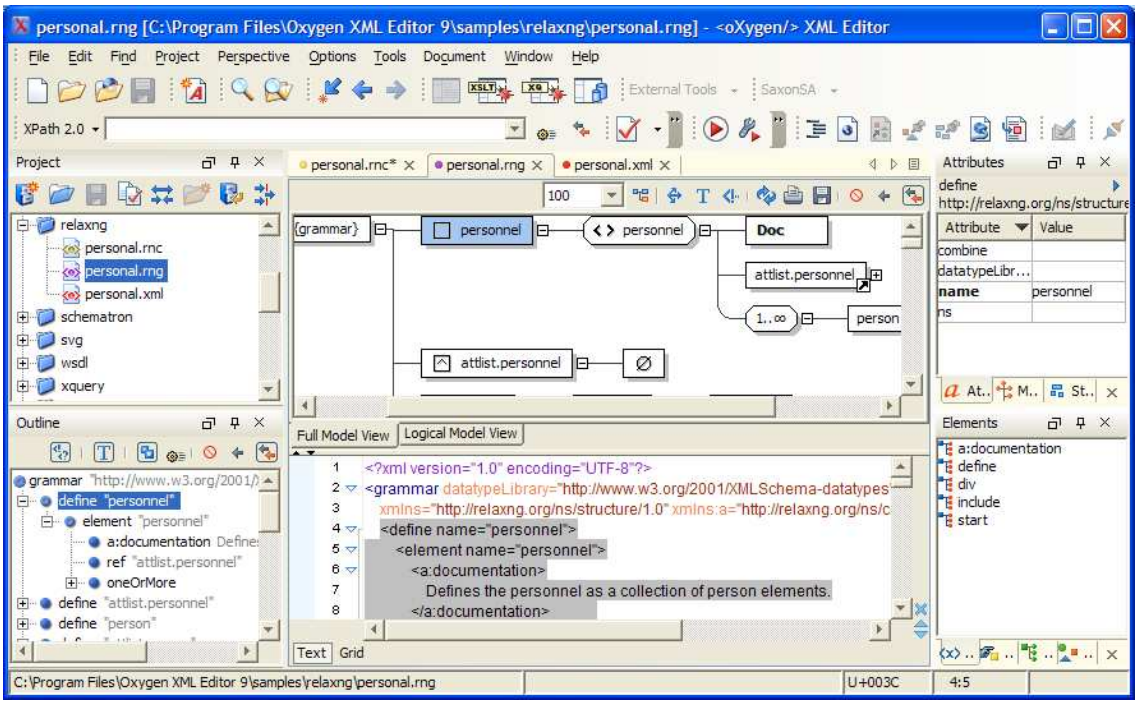
Formát inštrukcie však nie je najvhodnejší, pretože obsahuje aj názov softvéru a je tak zaručene nekompatibilný s konkurenčnými produktami. Prospešnejšie by bolo, keby sa tvorcovia známejších editorov dohodli na spoločnom formáte a vytvorili tak de-facto prenosný štandard.

Vzájomná konverzia schém medzi Relax NG, XML Schema, DTD a inštanciami XML je na dobrej úrovni, no u niektorých konštrukcií unikátnych pre daný jazyk si vyžaduje manuálny zásah.

Za nadštandardný sa dá považovať vizuálny editor, ktorý poskytuje stromový i názornejší diagramový pohľad, ktoré sa pri úpravách kódu obnovujú v reálnom čase a ponúkajú tak dokonalý prehľad o štruktúre schémy.



Obr. 2: oXygen XML Editor – Kompaktná syntax



Obr. 3: oXygen XML Editor - Diagram schémy

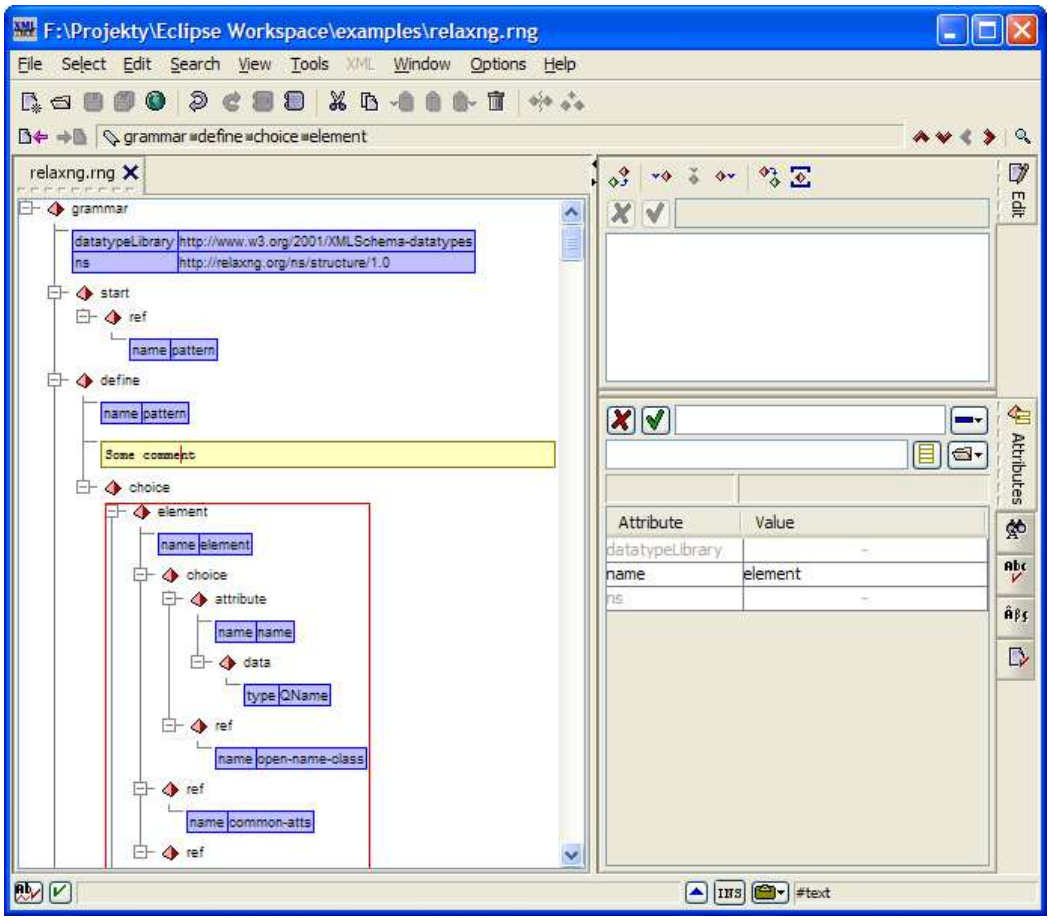
4.4 XMLmind 3

Podpora schém Relax NG je zahrnutá iba v platenej verzii programu. Nástroj umožňuje vizuálne upravovať DOM (Document Object Model) stromy Relax NG dokumentov v XML formáte, no zdrojové kódy je možné upravovať iba pomocou externých aplikácií. Súčasná verzia nedovoľuje priradiť k XML dokumentom príslušnú RNG schému, čím odpadá kontextovo-senzitívna asistencia pri ich úprave. Relax NG s kompaktnou syntaxou tiež nie je podporovaný. Používateľské rozhranie nie je príliš intuitívne.

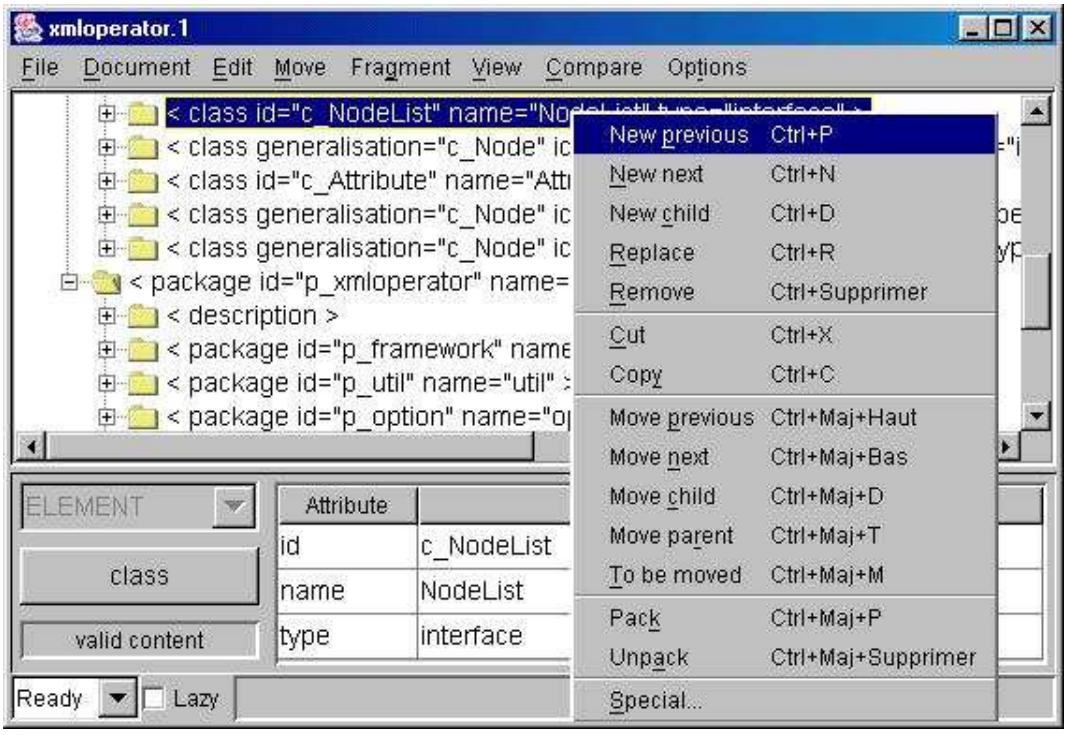
4.5 XMLOperator 3

Veľmi jednoduchý open-source nástroj na manipuláciu s XML stromami napísaný v jazyku Java. Pri úprave XML dokumentov poskytuje asistenciu na základe pridruženej Relax NG schémy. Autori deklarujú podporu rozsiahlych dokumentov a „neobmedzenú“ históriu úprav.

Ani táto aplikácia si však neporadí s kompaktnou syntaxou Relax NG.



Obr. 4: XMLMind Editor



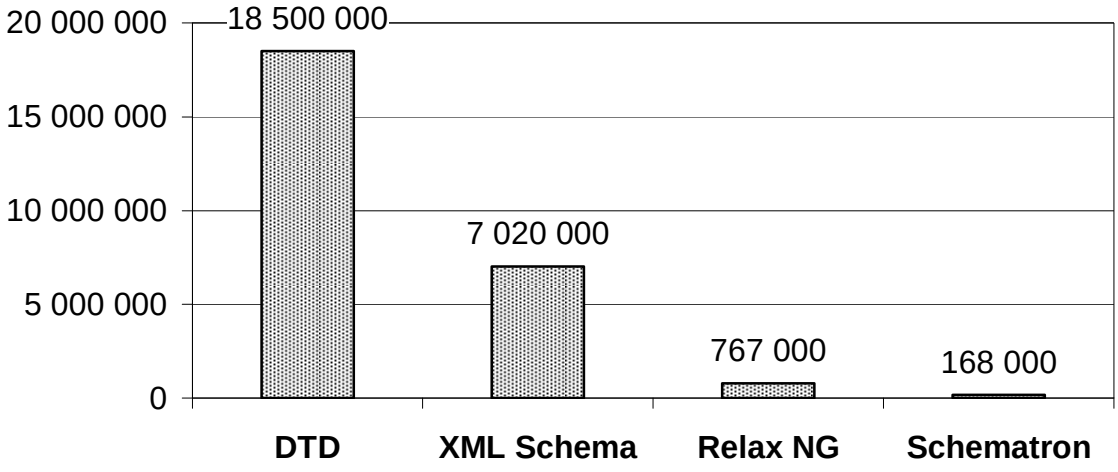
Obr. 5: XMLOperator

4.6 Topologi Collaborative Markup Editor 2.3

Za zmienku stojí ešte program Topologi Professional Edition určený na kolaboratívnu úpravu zdieľaných XML dokumentov. Jazyk Relax NG je síce podporovaný iba na úrovni validácie, no tá má niekoľko špecifických vlastností, ako je dávkové spracovanie, nastaviteľné filtre chybových hlášok podľa ich typu a závažnosti a podpora dátových typov z XML Schema. Je tak mimoriadne vhodný na správu rozsiahlych katalógov XML dokumentov v rozličných fázach publikačného procesu. Posledná verzia si už poradí aj s kompaktnou syntaxou.

4.7 Zhodnotenie

Jazyk Relax NG medzi vývojármi nie je zďaleka tak rozšírený ako jeho konkurencia, jazyky DTD a XML Schema. Je to logický dôsledok toho, že tieto pochádzajú z dielne konzorcia W3C, tvorcu samotného XML štandardu. Rozdielnu používanosť jednotlivých jazykov na definíciu XML schém potvrdzujú aj počty výsledkov fulltextového vyhľadávania ich názvov v internetovom obsahu pomocou služby Google, viď. nasledujúci graf.



Obr. 6: Počet zmienok o XML schémach na webových stránkach

Od toho sa potom odvíja aj miera jeho podpory v rôznych vývojárskych nástrojoch. Je však vidno, že do niekoľkých kvalitných komerčných aplikácií špecializovaných na XML si jazyk Relax NG už svoju cestu našiel a jeho podpora v nich nezaostáva za inými jazykmi.

Jazyk Relax NG sa pomaly dostáva aj do open-source aplikácií, no bude trvať ešte roky, kým sa výraznejšie presadí. Dobrým signálom je, že na portáloch hostujúcich vývoj open-source aplikácií sa objavujú desiatky nových projektov, ktorých autori deklarujú podporu Relax NG. Všetky sa ale nachádzajú v ranom štádiu vývojového cyklu.

Bude zaujímavé sledovať, ako ovplyvní rozšírenosť jazyka Relax NG očakávané dokončenie jeho štandardizácie na pôde ISO.

Kapitola 5

Editor Rexed - Dokumentácia

5.1 Popis programu

ReXeD je zásuvný modul (plug-in) pre populárne vývojové prostredie Eclipse slúžiaci na úpravu Relax NG a XML dokumentov. K jeho hlavným funkciám patria:

- farebné zvýrazňovanie kódu
- označovanie párových zátvoriek v kompaktnej syntaxi
- doplňovanie kódu do XML na základe schémy v Relax NG
- úprava dokumentov v stromovej reprezentácii
- konverzia schém medzi Relax NG s XML syntaxou, Relax NG s kompaktnou syntaxou a DTD

5.2 Systémové požiadavky

Editor pre svoju správnu funkčnosť potrebuje behové prostredie Java SE verzie 1.5 a vyššej a jeden z operačných systémov podporovaných platformou Eclipse. Z technických dôvodov bol plug-in testovaný iba na nasledovnej konfigurácii:

- Eclipse 3.2.2
- Microsoft Windows XP SP2
- Sun Java SE 6 Update 4
- architektúra Intel x86

5.3 Spustenie

Inštalácia nie je nutná, aplikáciu je možné spustiť priamo z jej umiestnenia („rexed/rexed.exe“). Editor je distribuovaný so všetkými knižnicami potrebnými k jeho správnej funkcionalite, vrátane minimálnej nutnej podmnožiny balíčkov platformy Eclipse.

Alternatívne je možné inštalovať samostatný zásuvný modul do príslušného adresára „plugins“ iného produktu postavenom na platforme Eclipse.

Pri prvom spustení je nutné zvoliť adresár (Eclipse Workspace), do ktorého bude Eclipse ukladať projekty a nastavenia. Súčasťou distribúcie je adresár

„rexed/workspace“, ktorý obsahuje ukážkové XML a Relax NG súbory. Tento adresár je samozrejme možné kedykoľvek zmeniť voľbou „File -> Switch Workspace…”.

5.4 Používateľské rozhranie

Okno aplikácie je rozdelené do niekoľkých častí:

Editor

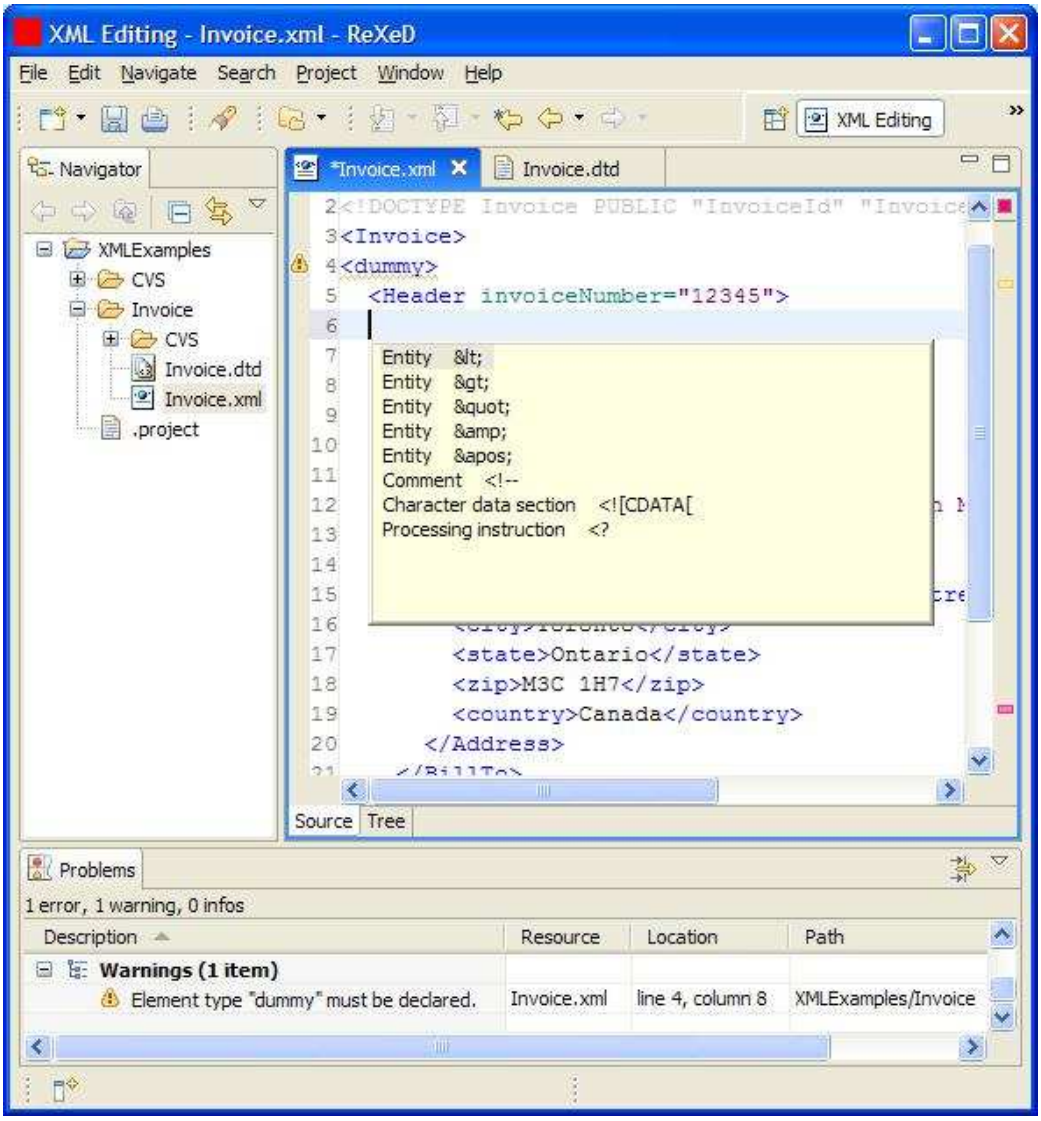
Slúži na prehliadanie a úpravu XML súborov. Obsah súboru môže byť zobrazený v textovej (karta Source) alebo stromovej (karta Tree) podobe. Pomocou lišty záložiek sa dá prepínať medzi jednotlivými otvorenými dokumentmi.

Navigator

V paneli Navigator je zobrazený obsah adresára Elipse Workspace. Umožňuje otváranie a manipuláciu so súbormi. Cez kontextové menu sú prístupné aj voľby „Convert to“ na konverziu schém do iných formátov a „Properties“ na priradenie schémy k XML dokumentu.

Problems

Upozornenia na prípadné syntaktické chyby v upravovaných dokumentoch sú zobrazované v podokne Problems. Po kliknutí na konkrétne upozornenie sa v Editore zvýrazní chybný úsek príslušného dokumentu.



Obr. 7: Používateľské rozhranie počas úpravy dokumentu

```
1<?xml version="1.0" encoding="UTF-8"?>
2<!DOCTYPE Invoice PUBLIC "InvoiceId" "Invoice.dtd" >
3<!-- this is some comment -->
4<Invoice>
5  <Header invoiceNumber="12345">
6    <Date>
7      <Month><![CDATA[ <July> ]]></Month>
8      <Day>15</Day>
9      <Year>2001</Year>
10   </Date>
```

Obr. 8: Farebne zvýraznený zdrojový kód

5.5 Funkcie editoru

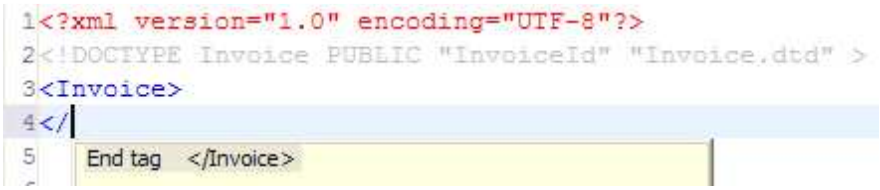
5.5.1 Zvýrazňovanie zdrojového kódu

Kvôli lepšej orientácii v prezeranom resp. upravovanom XML dokumente sú v ňom farebne odlíšené jednotlivé syntaktické konštrukcie. Editor rozpoznáva tieto:

- Element
- Atribút
- Hodnota atribútu
- Inštrukcia procesora
- Komentár
- Sekcia definície typu dokumentu (DOCTYPE)
- Sekcia znakových dát (CDATA)
- Obyčajný text

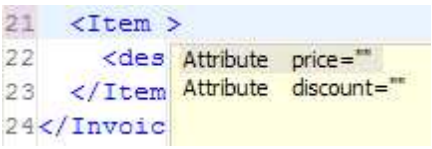
5.5.2 Doplňovanie kódu

Asistencia značne urýchľuje písanie kódu uzatváraním otvorených XML tagov, komentárov, inštrukcií procesora a `CDATA` sekcií. Ak je navyše k dokumentu

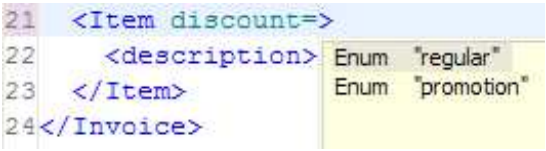


Obr. 9: Uzatvorenie tagu

Obr. 10: Dopĺňanie tagov



Obr. 11: Dopĺňanie atribútov



Obr. 12: Dopĺňanie hodnôt atribútov

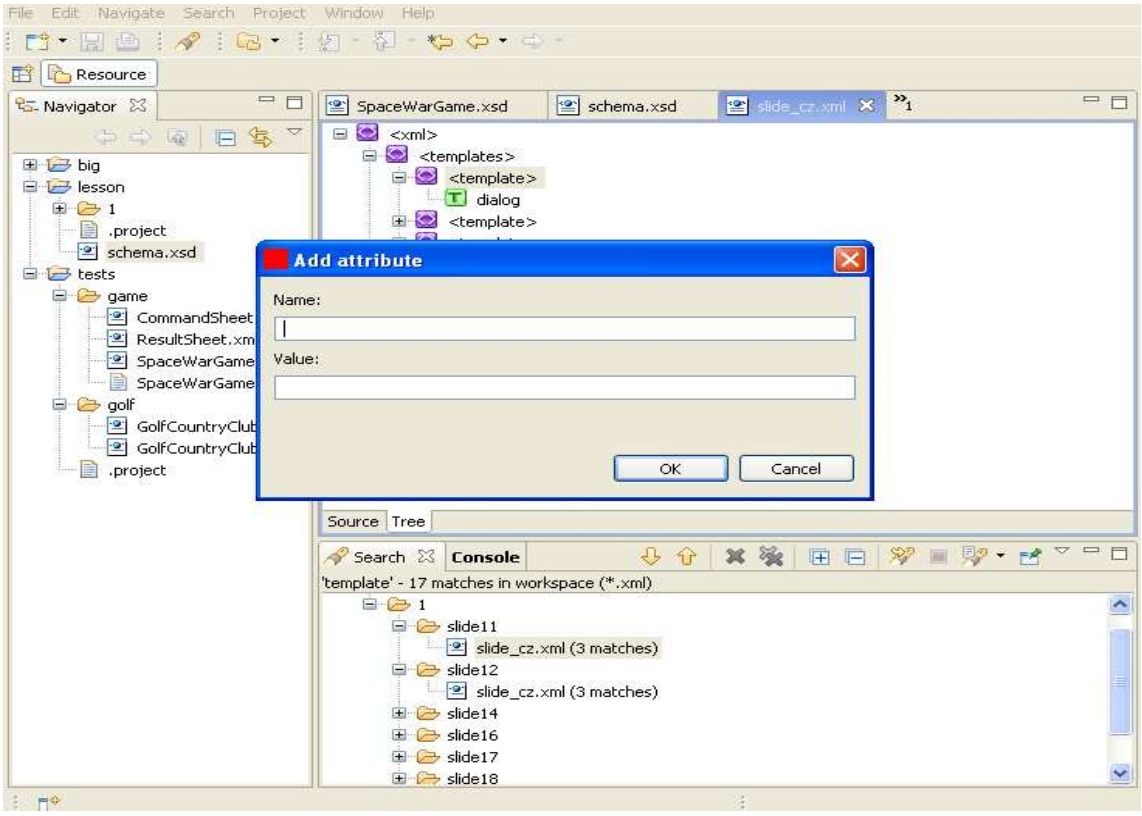
pridružený Relax NG súbor, vie editor v závislosti na aktuálnom kontexte ponúknuť zoznam prípustných elementov, entít, atribútov a ich hodnôt. Na priradenie schémy k dokumentu slúži dialógové okno.

Kontextová asistencia sa zobrazuje automaticky pri udalostiach ako je otvorenie tagu a vkladanie nového atribútu či entity. Dá sa však vyvolať aj manuálne cez voľbu „Edit -> Code Assist" alebo klávesovou skratkou CTRL+SPACE, ktorú je možné zmeniť v nastaveniach.

5.5.3 Stromový editor

Súčasťou programu je aj vizuálny editor, ktorý umožňuje manipuláciu so stromovou štruktúrou dokumentu, teda DOM modelom.

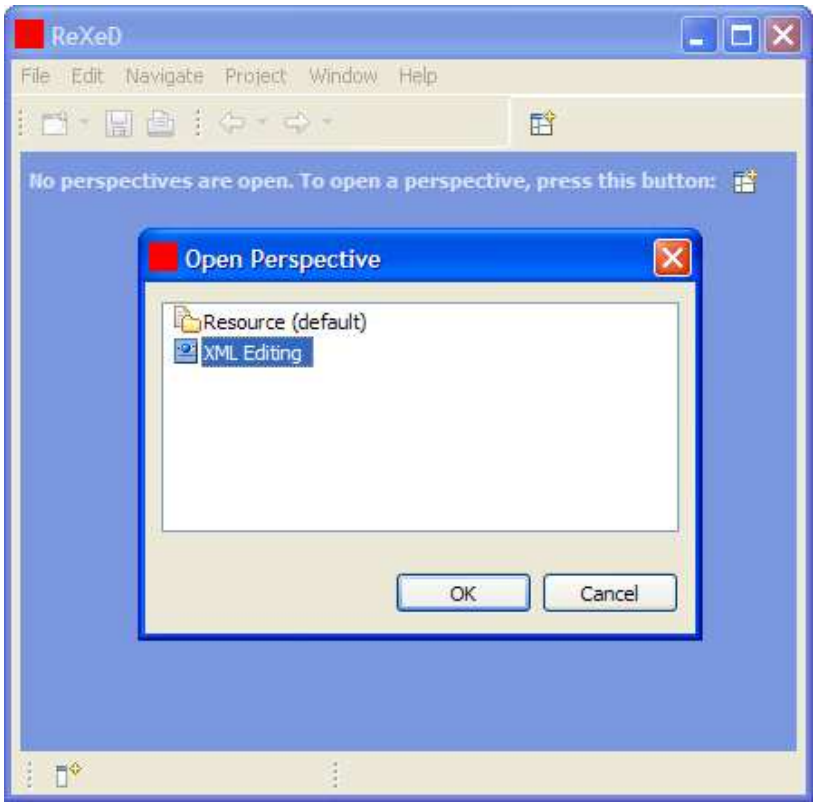
Po dvojkliku na element sa zobrazí dialógové okno s vlastnosťami daného elementu. Elementy sa dajú v štruktúre presúvať klasickým mechanizmom Cut&Paste.



Obr. 13: Stromový editor

5.5.4 Perspektíva

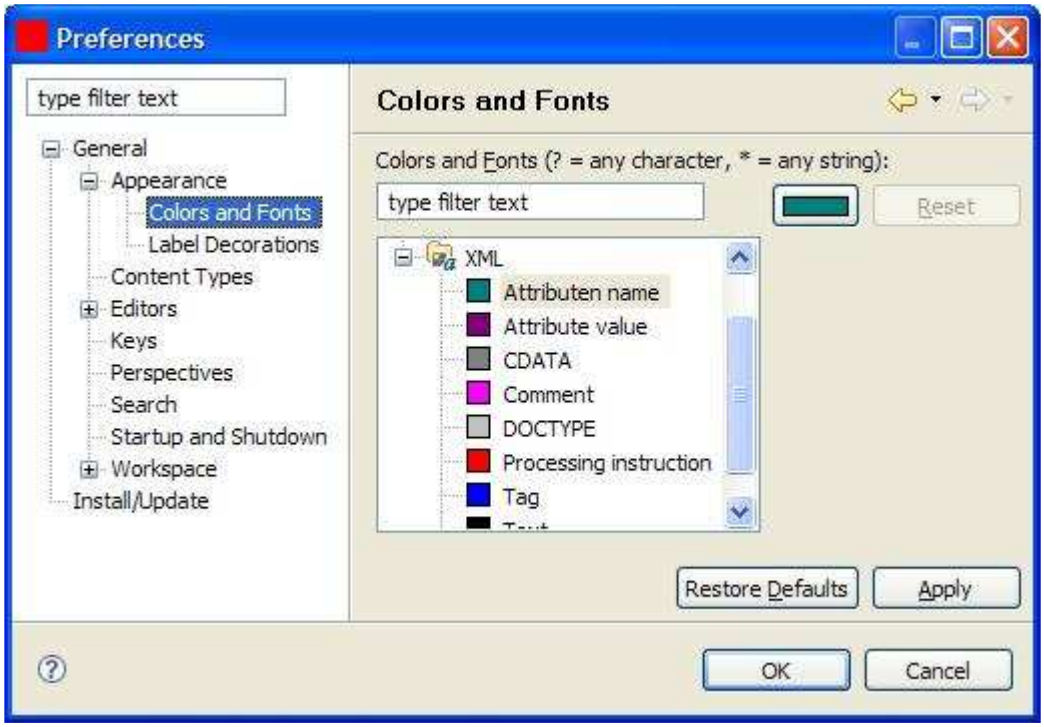
Zásuvný modul ReXeD pridáva do prostredia Eclipse perspektívu „XML Editing“, ktorá z východzieho používateľského rozhrania odstraňuje všetky nástroje, ktoré nesúvisia s úpravou XML dokumentov a tým zvyšuje prehľadnosť aplikácie. Medzi rôznymi perspektívami sa dá prepínať voľbou „Window -> Open Perspective“.



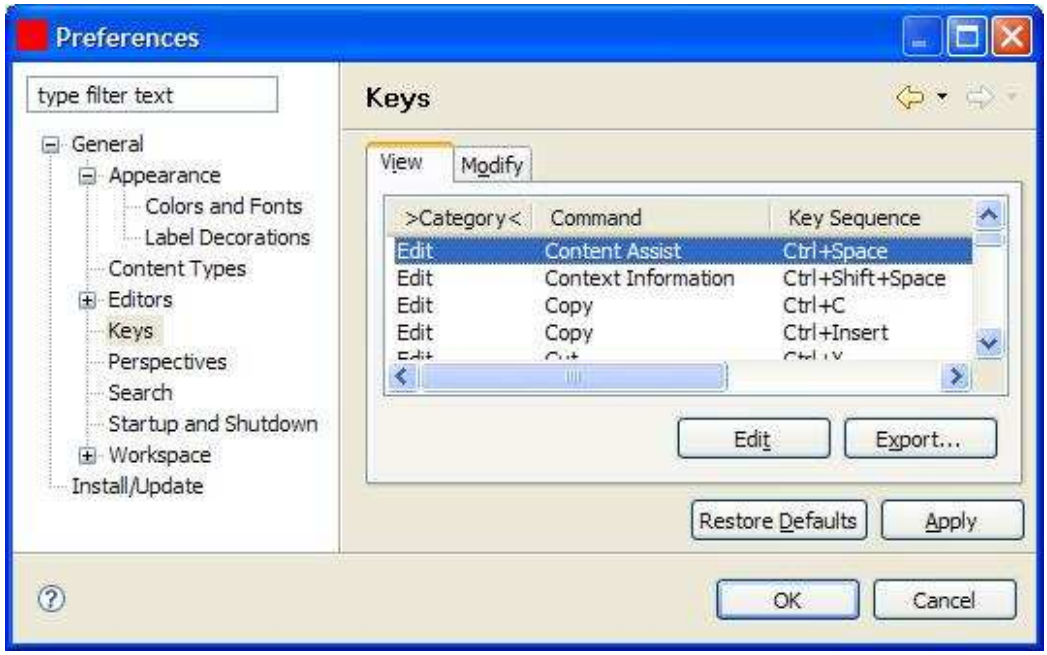
Obr. 14: Výber perspektívy

5.5.5 Nastavenia

Všetky nastavenia prostredia sú prístupné cez dialógové okno Preferences (voľba „Window -> Preferences...“). Program napríklad umožňuje prispôsobiť farby, ktorými sa zvýrazňujú jednotlivé časti XML kódu a zmeniť klávesovú skratku, ktorá vyvoláva kontextového asistenta.



Obr. 15: Nastavenie farieb



Obr. 16: Nastavenie klávesových skratiek

Kapitola 6

Editor Rexed - Implementácia

6.1 Výber platformy

Platforma Eclipse bola zvolená najmä preto, že je vďaka použitiu jazyka Java nezávislá na operačnom systéme a disponuje robustným mechanizmom rozšíriteľnosti. Väčšina funkcionality editoru sa do nej dá pridať implementáciou vstavaných rozhraní, ktoré poskytujú vysokú mieru abstrakcie a vyznačujú sa kvalitným objektovým návrhom. Vysoká modularita je dosiahnutá aj implementáciou štandardu OSGi.

6.2 Komponenty

6.2.1 Plug-in

Pri načítaní modulu sa vytvorí jednoduchý singleton triedy `RexedPlugin`, ktorá je rozšírením triedy `AbstractUIPlugin`. Slúži na prístup k zdrojom zdieľaným všetkými inštanciami editora.

6.2.2 Editor

Od triedy `MultiPageEditorPart` odvodený `XMLEditor` sa stará o otváranie a ukladanie dokumentov. Samotné upravovanie prenecháva zapuzdreným objektom `XMLEditorSource` a `XMLEditorTree`, ktorých inicializáciu a prepínanie medzi nimi pomocou kariet má tiež na starosti.

Balíček `org.eclipse.ui.editors.text` poskytuje základnú implementáciu textového editoru `TextEditor`, ktorú trieda `XMLEditorSource` rozširuje o možnosť validácie, zvýrazňovania a doplňovania kódu. Ku každému editoru je priradený objekt typu `XMLSourceViewerConfiguration`, ktorý má na starosti pridanie spomínanej funkcionality.

6.2.3 Zvýrazňovanie kódu

Najprv rozdelí `XMLPartitionScanner` dokument podľa obsahu (napríklad tag, komentár a pod.) na menšie neprekrývajúce sa úseky, nazývané partície, ktoré sú definované v triede `XMLDocumentProvider`. Každému typu partície je potom pridelený

špecifický objekt s rozhraním ITokenScanner. Jedná sa o konečný stavový automat, ktorý na základe vstupného prúdu znakov mení aktuálnu farbu textu.

Prefarbovanie textu je optimalizované tak, aby sa pri každej lokálnej zmene textu nemusel zneplatniť celý obsah, čo by bolo veľmi neefektívne, najmä u väčších dokumentov. Niektoré zmeny, ako napríklad vloženie začiatku komentára, si ale môžu vyžiadať opätovné zafarbenie textu od aktuálnej pozície až do konca dokumentu.

6.2.4 Doplňovanie kódu

Každému typu partície je pridelený jeho vlastný kontextový asistent s rozhraním `IContentAssistProcessor`, ktorý v prípade automatického alebo vynúteného zavolania metódy `computeCompletionProposals` vráti zoznam slov, ktoré sa môžu vyskytovať v okolí aktuálneho umiestnenia karetu v dokumente. Je potom na používateľovi, ktorú z ponúk akceptuje.

Každé z týchto ponúkaných slov je obalené v triede ExtendedCompletionProposal, ktorá v dokumente nájde prípadný prefix slova pred kurzorom a posunie ofset na správne miesto. Rovnako sleduje aj následne vkladané znaky a anuluje sa, ak sa prefix prestane zhodovať so začiatkom slova.

Mimo tagov je asistencia jednoduchá. Ak sa karet nachádza na začiatku dokumentu, je ponúknuté vloženie XML deklarácie. V sekciách CData, komentároch a inštrukciách spracovnia je zas ponúkaná iba ukončovacia sekvencia danej partície.

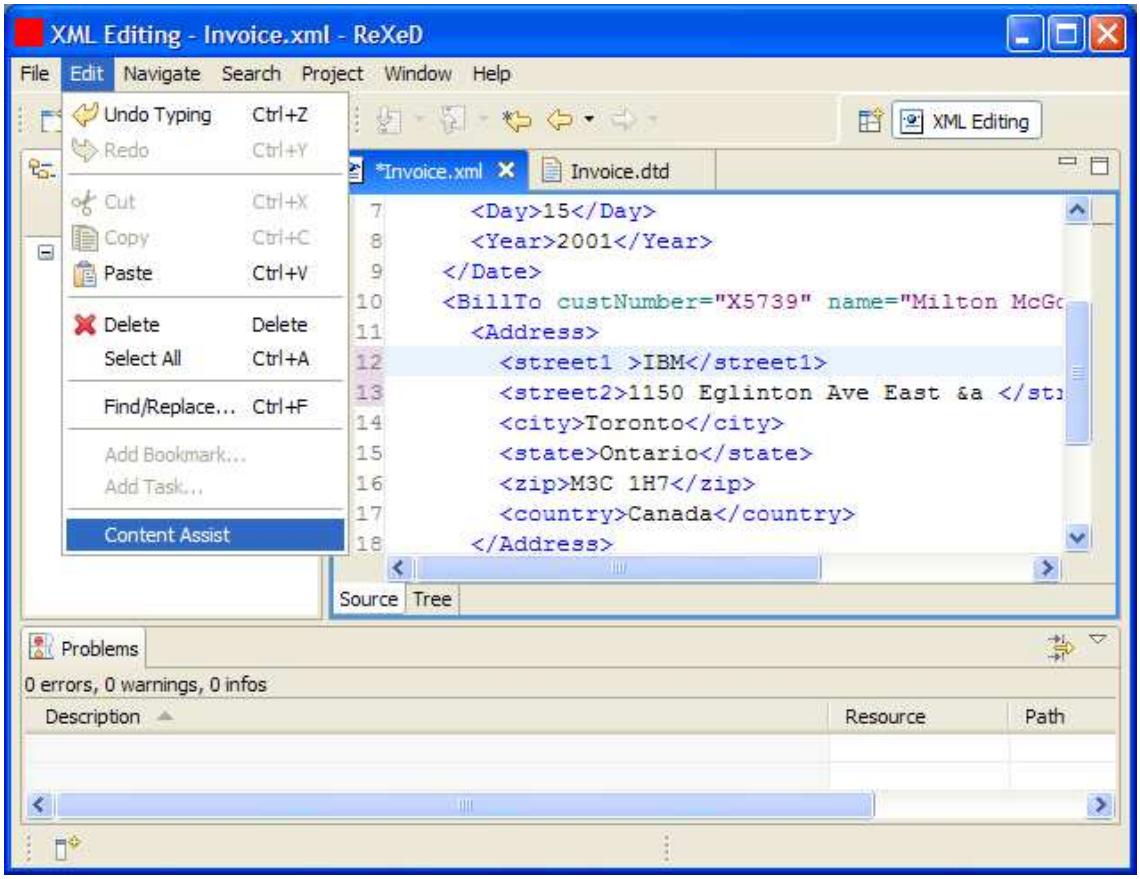
Pri dopĺňaní ukončovacieho tagu, t.j. za sekvenciou “</”, je nutné zistiť názov párového otváracieho tagu. Ten sa zistí automatom, ktorý prechádza cez predchádzajúce ukončovacie a začiatočné tagy a udržiava si rozdiel ich počtu, pričom ignoruje prázdne tagy. Hľadaný otvárací tag je ten, pred ktorého spracovaním rozdiel nulový. Názov elementu je potom reťazec medzi znakom ‘<’ a ľubovoľným bielym znakom.

Ostatné funkcie sú už závislé na pridruženom Relax NG. Na určenie kontextu aktuálneho elementu sa opäť použije zásobníkový automat prehľadávajúci strom až ku koreňu. Získaná informácia sa potom použije na zostavenie ponuky prípustných elementov. Pri doplňovaní názvu atribútu sa zistí názov aktuálneho elementu (viď predošlý odsek) a v prípade, že je definovaný, dohľadá sa v Relax NG množina jeho prípustných atribútov. Obdobne, k získaniu zoznamu možných hodnôt atribútu je nutné poznať jeho názov a element, ku ktorému patrí. Hodnoty je možné určiť iba v prípade, ak je obsah definovaný ako vymenovaný typ, pretože tieto sú priamo uvedené v Relax NG dokumente.

6.2.5 Príkazy

Príkazy sú v terminológii Eclipse sémantické deklarácie chovania (napr. Vložiť), na ktoré sa môžu napojiť jednotlivé prvky používateľského rozhrania (napr. položky menu, tlačítka či klávesové skratky). S príkazmi sú potom asociované akcie, ktoré tieto príkazy vykonávajú (napr. vloženie textu zo schránky do aktuálneho dokumentu).

Objekt `XMLEditorContributor` pridáva do menu možnosť vyvolať kontextového asistenta. Keď niektorý zo zapuzdrených editorov XML dostane focus, `XMLEditorContributor` od neho cez rozhranie `IActionProvider` zistí, ktoré akcie poskytuje a presmeruje na ne príslušné príkazy.



Obr. 17: Príkazy v menu

6.2.6 Perspektíva

Za vzhľad aplikácie, t.j. rozloženie okien, tlačítok a ďalších prvkov rozhrania zodpovedá trieda `XMLPerspective`. Pri otvorení perspektívy na úpravu XML dokumentov sa zavolá metóda `createInitialLayout`, ktorá do východzej prázdnej perspektívy pridá potrebné objekty.

6.2.7 Konverzia medzi formátmi

Do kontextového menu súborov typu `*.rnc`, `*.rng` a `*.dtd` je deklaratívne (t.j. cez súbor `plugin.xml`) pridaná položka „Convert to“ obsahujúca jednotlivé konverzné príkazy. Volanie príkazov je potom delegované na metódu `run()` príslušných potomkov abstraktnej triedy `ConvertAction`. Na samotnú konverziu sa používa knižnica `Trang`. Výstup je následne uložený na rovnaké umiestnenie ako vstupný súbor, zmení sa iba koncovka.

Knižnica žiaľ nedisponuje rozhraním pre prúdový vstup/výstup a manipuluje so súbormi sama. Konverzia tak nefunguje na iných ako lokálnych súborových systémoch a po jej ukončení je nutné manuálne obnoviť zobrazený obsah adresára.

6.2.8 XML Strom

Na zobrazenie XML stromu slúži samostatná kompomenta `XMLEditorTree` [19]. Pri prepínaní sa medzi zdrojovým a stromovým pohľadom je nutné zabezpečiť synchronizáciu ich obsahu metódou `XMLEditor.pageChange()`. Komponenta podporuje iba XML syntax, takže v prípade úpravy schémy v kompaktnej syntaxi je nutné vykonávať pri synchronizácii aj konverziu medzi formátmi.

6.2.9 Nastavenie farieb

Preddefinované farby jednotlivých častí XML sú nastavené deklaratívne v súbore „build.xml“. Každé okno editora je zaregistrované na sledovanie zmien vlastností (`PropertyChangeEvent`) a ak sa zmena týka farby používanej editorom, sú na ňu upozornené všetky objekty `ITokenScanner` starajúce sa o zafarbovanie textu. Následne je prezentácia zneplatnená a text prekreslený s novou farebnou schémou.

6.2.10 Lokalizácia

S ohľadom na možnú budúcu lokalizáciu sú externalizované a centralizované všetky reťazce, ktoré sa zobrazujú v používateľskom rozhraní. K týmto reťazcom sa pristupuje jednotne cez položky triedy `cz.cuni.mff.rexed.localization.Localization`. Samotné lokalizovateľné reťazce sú uložené v súbore `Localization.properties`.

Nová lokalizácia sa vytvorí zduplikovaním tohto súboru a pridaním identifikátora daného jazyka pred príponu. Napríklad súbor s nemeckým prekladom musí mať názov `Localization_de_DE.properties`.

6.3 Možnosti rozšírenia

Medzi editorom zdrojového kódu a stromovej štruktúry by mohla byť väčšia previazanosť, ktorá by dovolila napríklad zachovanie kontextu pri prepínaní sa medzi nimi alebo zobrazenie oboch editorov naraz.

Kontextová asistencia má svoje medzery a v niektorých prípadoch zobrazuje viac elementov, ako na danom mieste povoľuje gramatika. Je to spôsobené optimalizáciou analýzy, ktorá sa pri vyvolaní kontextového asistenta musí javiť ako okamžitá. Náprava by si vyžadovala implementáciu vlastného validátora Relax NG, ktorý by v aktuálnom kontexte namiesto prípustnosti konkrétneho elementu či atribútu vrátil zoznam všetkých prípustných elementov či atribútov.

Plug-in pridáva do prostredia Eclipse aj vlastný XML editor, v ktorom umožňuje dopĺňanie kódu na základe pridruženej Relax NG schémy. Vhodnejšie by bolo integrovať ho s oficiálnym modulom Web Standards Tools [15], ku ktorému ale zatiaľ chýba kvalitná programátorská dokumentácia.

Možností vylepšenia je však ako u každého softvérového diela neobmedzene mnoho a preto sa nedajú vymenovať všetky.

Kapitola 7

Záver

Cieľom práce bolo priblížiť problematiku editorov Relax NG a vytvoriť vlastný editor s pokročilými funkciami.

Editor Rexed, ktorý vznikol v rámci písania práce, značne urýchľuje vývoj schém v jazyku Relax NG najmä vďaka farebnému zvýrazňovaniu kódu, kontextovej asistencii a stromovému pohľadu. Túto podporu pridáva ako zásuvný modul do prostredia Eclipse, čím zvyšuje jeho použiteľnosť v oblasti editácie XML, ktorá má pred sebou ešte veľký potenciál.

Za masovým rozšírením väčšiny počítačových jazykov stojí aj dostupnosť kvalitných nástrojov, ktoré umožňujú rapídnu úpravu kódu a sú podľa možností bezplatné. Preto aj v tejto práci popísaný editor môže svojim dielom prispieť k obľúbenosti jazyka Relax NG.

Príloha A

Obsah CD-ROM

Súčasťou bakalárskej práce je priložené CD-ROM, ktoré obsahuje zdrojové kódy editoru, ukážkové XML a Relax NG súbory. Adresárová štruktúra je nasledovná:

- rexed/ - spustiteľný editor
- src/ - zdrojové kódy programu
- doc/ - vygenerovaná dokumentácia Javadoc
- jre/ - inštalátor Java 6 SE
- grafnetter.pdf – text práce v elektronickej podobe

Zoznam obrázkov

Použitá literatúra

[1] Dokumentácia k Relax NG, http://relaxng.org/

[2] Mlýnková, I. - Pokorný, J. - Richta, K. - Toman, K. - Toman, V.: Technologie XML. Skripta. Karlova Univerzita, Praha, Česká republika, 2006.

[3] McLaughlin, B.: Java and XML. O'Reilly & Associates, California, USA, 2000

[4] Eckel, B.: Thinking in Java 2nd Edition. Prentice Hall, New Jersey, USA, 2000

[5] Eclipse SDK Help, http://help.eclipse.com

[6] Clayberg, E. – Rubel, D.: Eclipse: Building Commercial-Quality Plug-ins, Second Edition. Addison Wesley Professional**,** USA, 2006

[7] Daum, B.: Java-Entwicklung mit Eclipse 3, dpunkt.verlag GmbH, Heidelberg, SRN, 2004

[8] XML Schema: Formal Description, W3C Working Draft, 20 March 2001, http://www.w3.org/TR/2001/WD-xmlschema-formal-20010320/

[9] Extensible Markup Language (XML) 1.0 (Fourth Edition), http://www.w3.org/TR/REC-xml/

[10] Thompson, H. S. – Tobin, R.: Using Finite State Automata to Implement W3C XML Schema Content Model Validation and Restriction Checking, Edinburgh, UK, 2003

[11] Relax NG, Compared http://www.xml.com/lpt/a/2002/01/23/relaxng.html

[12] Clark, James: RELAX NG and W3C XML Schema, http://www.imc.org/ietf-xml-use/mail-archive/msg00217.html

[13] Relax NG DTD Compatibility, http://relaxng.org/compatibility-20011203.html

[14] van der Vlist, Eric: Relax NG, O'Reilly & Associates, California, USA, 2003

[15] Eclipse Web Standard Tools, http://www.eclipse.org/webtools/wst/main.php

[16] Runtime requirements for Eclipse, http://www.eclipse.org/downloads/moreinfo/jre.php

[17] Document Schema Definition Languages, http://www.dsdl.org

[18] Relax NG Specification, http://www.oasis-open.org/committees/relax-ng/spec-20011203.html

[19] Flídr, Jindřich: Správce lokalizací XML souborů, 2008